Model-based Dynamic Shielding for Safe and Efficient Multi-Agent Reinforcement Learning

Wenli Xiao
The Chinese University of Hong Kong, Shenzhen
Shenzhen Institute of Artificial Intelligence and Robotics for Society
China
wenlixiao@link.cuhk.edu.cn

Yiwei Lyu
Carnegie Mellon University
United States
yiweilyu@andrew.cmu.edu

John Dolan
Carnegie Mellon University
United States
jdolan@andrew.cmu.edu

ABSTRACT

Multi-Agent Reinforcement Learning (MARL) discovers policies that maximize reward but do not have safety guarantees during the learning and deployment phases. Although shielding with Linear Temporal Logic (LTL) is a promising formal method to ensure safety in single-agent Reinforcement Learning (RL), it results in conservative behaviors when scaling to multi-agent scenarios. Additionally, it poses computational challenges for synthesizing shields in complex multi-agent environments. This work introduces Model-based Dynamic Shielding (MBDS) to support MARL algorithm design. Our algorithm synthesizes distributive shields, which are reactive systems running in parallel with each MARL agent, to monitor and rectify unsafe behaviors. The shields can dynamically split, merge, and recompute based on agents' states. This design enables efficient synthesis of shields to monitor agents in complex environments without coordination overheads. We also propose an algorithm to synthesize shields without prior knowledge of the dynamics model. The proposed algorithm obtains an approximate world model by interacting with the environment during the early stage of exploration, making our MBDS enjoy formal safety guarantees with high probability. We demonstrate in simulations that our framework can surpass existing baselines in terms of safety guarantees and learning performance.

KEYWORDS

Robotics; Multi-Agent Reinforcement Learning; Safety

ACM Reference Format:
Wenli Xiao, Yiwei Lyu, and John Dolan. 2023. Model-based Dynamic Shielding for Safe and Efficient Multi-Agent Reinforcement Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

Multi-Agent Reinforcement Learning (MARL) [10, 59] is a promising approach to obtain learning control policies for multi-agent decision-making tasks such as transportation management [3, 40], motion control [42, 57], and autonomous driving [7, 47, 60]. However, applying MARL methods in safety-critical autonomous systems (e.g., autonomous driving cars) can cause havoc due to the lack of formal safety guarantees. In addition, traditional MARL approaches with behavior penalties (i.e., giving a negative reward for unsafe actions) cannot ensure safety in practice [18, 45]. Therefore, there is a significant challenge to developing safe MARL systems that are provably trustworthy [11, 18, 33, 45, 59].

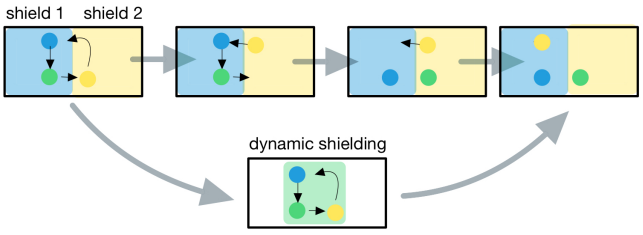


Figure 1: The differently colored circles denote multiple agents, and the black arrows are desired actions. Traditional decentralized shielding (upper) takes extra steps in waiting for coordination near the border of shields, while proposed dynamic shielding (lower) eliminates this overhead.

Recently, there has been much research in notions of safety [1, 18, 19, 24, 41]. For example, Linear Temporal Logic (LTL) [43] is a specification language used for formal verification to ensure that an automation system always stays in safe states [46]. A recent work [1] adopts LTL as a safety specification language in single-agent Reinforcement Learning (RL) via synthesizing a shield to monitor the RL agent. The shield is a lightweight system running along with the RL agent, which monitors actions selected by the RL agent and rejects any unsafe actions according to the given safety specification. The shield has provable safety guarantees for the lifetime of the RL process (i.e., the training and deployment phases). Factored shielding [18] adapts the shielded learning method to multi-agent scenarios in a decentralized fashion. Compared with centralized shielding, which uses one shield to monitor the states and actions of all agents, factored shielding synthesizes multiple shields, and each shield monitors a subset of the agents' state space. These methods perform well in discrete environments. However, both centralized shielding and factored shielding are challenging to scale up for more complex continuous environments.

On the other hand, when it comes to shielding framework design, there is a dilemma: centralized approaches have limited scalability [18], while fully decentralized methods cause coordination overheads. Agents can become stuck waiting for coordination when they get closer to one another due to the lack of information sharing

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

in decentralized approaches. For instance, Figure 1 shows a scenario in which factored shielding causes extra coordination overhead. In this paper, we propose a novel, safe, and efficient MARL framework in a mixed decentralized manner, which dynamically synthesizes shields to mitigate those limitations.

Specifically, our main contributions are threefold: Firstly, we propose a novel shield framework - *dynamic shielding*, which enables robots to collaborate to ensure safety. There are initially multiple shields, which concurrently monitor different agents. When there is a high risk of conservative behavior (e.g., agents move together), the shields could choose to merge with others. The merged shield can leverage the state information of multiple agents to mitigate unnecessary coordination overhead. When agents move apart from each other, the merged shield can split into multiple shields. We also present an effective shield synthesis approach in section 5, named *k-step look ahead shields*. Our method prunes the unnecessary computation of traditional shield synthesis approaches [1, 18] and delegates the computation complexity to the online algorithm, which can synthesize shields in real-time. We also incorporate a world-model learning procedure to learn a simplified environment dynamics model. This enables our framework to learn from scratch, with minimal external knowledge.

Additionally, we showcase the effectiveness and performance of our shielding approach through extensive experiments. We study the navigation problem on six different grid world maps [38] and two different tasks in the Multi-Agent Particle Environment [32] (MPE). Our approach outperforms other baselines in terms of reward and minimal steps while guaranteeing safety. Furthermore, we show that dynamic shielding ensures safety with a high probability as the number of agents scales up.

2 RELATED WORK

2.1 Safe Multi-Agent Reinforcement Learning

Safe RL methods can be classified into two categories [19]: 1) The first is optimization criterion-based methods, which modify the RL objective functions [15, 50, 53]. For example, SNO-MDP [54] tackles the safe RL problem using a constrained Markov decision process. 2) The second is based on modifying the exploration process to avoid undesirable actions [6, 26, 30], which incorporates extra domain-specific knowledge (e.g., and demonstration) into the training process. Our dynamic shielding algorithm falls into the second category. Shielding was introduced to RL in [1], and was adapted to multi-agent settings in [18]. In this work, we propose a novel shielding framework for MARL by addressing challenges such as coordination overhead and scalability issues in the multi-agent setting and mitigating the reliance on external knowledge.

2.2 Safe Control via Control Barrier Functions

Barrier certificates [34, 44] and Control Barrier Function (CBF) [56] based control methods [12, 14, 35, 36, 49, 52] are commonly used to provide safety guarantee for safety-critical problems, such as collision avoidance [37, 44, 56, 58]. In the context of multi-agent collision avoidance, previous research has explored using the multi-agent CBF frameworks [8, 37, 55]. Recent works [13, 45] have proposed decentralized controller synthesis approaches under the CBF that can scale to an arbitrary number of agents. We acknowledge their contributions, but they are perpendicular to our focus.

In this work, we aim to address more general safety specifications for MARL by leveraging a more expressive Linear Temporal Logic (LTL) [43]. LTL can conveniently capture complex time-varying constraints [48]. Although we conduct experiments for collision-avoidance tasks in section 6, we focus on the use of LTL for MARL in this work, with the aim of extending our approach to even more complex safety constraints in the future.

2.3 LTL as Safety Specification

LTL is a widely used specification language in safety-critical systems [2, 4], which can express complex requests at a high level. For example, LTL has been used to express complex task specifications for robotic planning and control [28, 51]. Several works [9, 23, 25] develop reward shaping techniques that translate logical constraints expressed in LTL to reward functions for RL. However, [18, 45] has empirically demonstrated reward shaping cannot ensure safety in MARL. Our shield synthesis technique, which is based on solving two-player safety games, was originally developed in [27] to enforce LTL specifications. The original technique synthesizes shields to local caches in an offline manner. In section 5, we propose a novel online method to synthesize shields in real time.

3 PRELIMINARIES

We start by introducing *Multi-Agent Reinforcement Learning*, *Shielding*, and *Safety Games with Linear Temporal Logic specification*, upon which our algorithm builds.

3.1 Multi-Agent Reinforcement Learning

We focus on the n-player Markov Games defined by a tuple

$$\left(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{r^i\}_{i\in\mathcal{N}}, \mathcal{P}, \gamma\right)$$

where $\mathcal{N} = \{1...n\}$ is the set of n agents, $\mathcal{S}$ denotes the state space jointly observed by all agents, $\mathcal{A}^i$ is the action space of agent i, r^i is the reward function of agent i, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition probability, and γ is the discount factor. We assume the initial state s_1 follows a fixed distribution $\rho \in \Delta(\mathcal{S})$. At each time step t, the agents observe state s_t, take actions $a_{t,i} \in A^i$ in the environment simultaneously, and receive rewards $r_{t,i} \in R^i$. Then the state of the environment moves to s_{t+1}. The objective of each agent i is to learn a control policy π_i which maximizes the expected cumulative reward $E\left[\sum_{t=0}^{\infty} \gamma^t R^i\left(s_t, a_t, s_{t+1}\right)\right]$. MARL algorithms can be categorized into three different types based on the dependence of individual agent performance on other agents' choices, including cooperative, competitive, and mixed settings. We use MARL algorithms with mixed settings in our experiment in Section 6. CQ-learning [16] is a MARL algorithm that enables agents to behave separately at most of the time and consider the states and actions of other agents when necessary. MADDPG [32] is a deep MARL algorithm with centralized training and decentralized execution, each agent trains models that simulate each of the other agents' policies based on its observation of their behavior.

3.2 LTL as Safety Specification

We consider Linear Temporal Logic [43] (LTL) to express safety specifications. LTL is an extension of propositional logic, which has long been used as a tool in the formal verification of programs and systems. The syntax of LTL is given by the following grammar [5]:

$$\varphi := p | \neg p | \varphi_1 \vee \varphi_2 | \bigcirc \varphi | \varphi_1 \mathcal{U} \varphi_2$$

where p is an atomic proposition. The temporal operators are next $\bigcirc\varphi$, which indicates φ is true in the next succeeding state, and until $\varphi_1 \mathcal{U} \varphi_2$ indicating φ_1 is true until the state where φ_2 is true. From these operators, we can define $True \equiv \phi \vee \neg\phi$, $False \equiv \neg True$, implication $\varphi \Rightarrow \psi := \neg\varphi \vee \psi$, eventually $\diamond\varphi := \text{True}\,\mathcal{U}\varphi$, and always $\square\varphi := \neg \diamond \neg\varphi$. We use LTL formulas to express safe specifications. For example, $\square\neg collision$ denotes that collision should never happen. We consider translating the LTL safety specification into a safe language accepted by a deterministic finite automaton (DFA) [29]. In addition, we extend the definition of safe RL in [1] to MARL in the following way:

Definition 1. *Safe MARL is the process of learning optimal policies for multiple agents while satisfying a temporal logic safety specification ϕ^s during the learning and execution phases.*

3.3 Formal Safety Guarantee with Shield

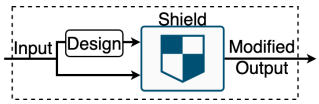


Figure 2: Enforce safety specification via shielding.

Our method builds upon a prior method called Shield [18, 27], which ensures safety properties at runtime. A Shield (shown in Figure 2) monitors the control input of agents and corrects any unsafe control input instantaneously. A Shield should have two properties: 1) Minimal interference. Namely, shields only correct the action if it violates the safety rule. 2) Correctness. Shields should distinguish every unsafe action and refine it with safe actions. Our method uses the Shield framework to ensure safety, and we provide theoretical proof of safety in section 5.

We **represent the shield** using a finite-state reactive system. According to the formulation in [18], a finite-state reactive system is a tuple $S = (Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$, where Σ_I and Σ_O are the I/O alphabets, Q is the state set, $q_0 \in Q$ denotes the initial state, $\delta : Q \times \Sigma_I \to Q$ is a transition function, and $\lambda : Q \times \Sigma_I \to \Sigma_O$ is an output function. Given the symbolic abstraction of the control input (i.e., input trace) $\overline{\sigma_I} = x_0 x_1 \ldots \in \Sigma_I^\infty$, the system S generates the trajectory of states (i.e., output trace) $\overline{\sigma_O} = \mathcal{S}(\overline{\sigma_I}) = \lambda(q_0, x_0)\,\lambda(q_1, x_1) \ldots \in \Sigma_O^\infty$, where $q_{i+1} = \delta(q_i, x_i)$ for all $i \geq 0$.

We **synthesize the shield** by solving a *two-player safety game* [27], a game played by the MARL agents and the environment, where the winning condition is defined by the LTL safety specification. MARL agents should comply with all safety specifications all of the time in order to win the game. A two-layer game is a tuple $\mathcal{G} = (G, g_0, \Sigma_I, \Sigma_O, \delta, win)$ with a finite set of game states G, the initial state $g_0 \in G$, a complete transition function $\delta : G \times \Sigma_I \times \Sigma_O \to G$, and *win* as a winning condition. In every state $g \in G$, the environment first chooses an input action $\sigma_I \in \Sigma_I$, and then the MARL agents choose a joint action (in abstraction symbol) $\sigma_O \in \Sigma_O$. Then the game moves to the next state $g' = \delta(g, \sigma_I, \sigma_O)$, and so forth. The resulting trajectory of game states $\bar{g} = g_0, g_1, \ldots$ is called a *play*. A play is won if and only if $win(\bar{g})$ is *true*. We describe the detailed procedure of synthesizing shields via solving the two-player safety game in section 5.

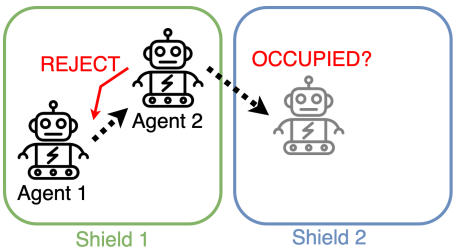


Figure 3: The green and blue squares denote shields, and the dashed arrows are desired actions of agents. There is no communication between shields 1 and 2. Therefore, Shield 1 conservatively judges that agent 2 cannot successfully enter shield 2, thus rejects Agent 1's action.

4 TACKLING SAFE AND EFFICIENT MULTI-AGENT REINFORCEMENT LEARNING VIA DYNAMIC SHIELDING

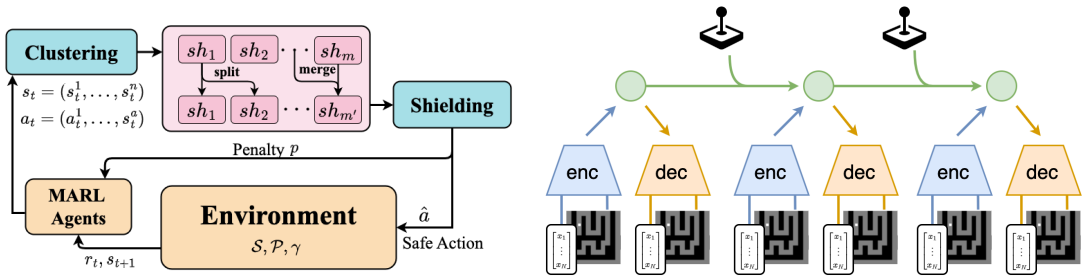
In this section, we first describe how traditional shielding methods cause sub-optimal learning. Then, we present our method for safe, optimal, and optimal MARL learning.

4.1 Conservative Behavior and Coordination Overhead

For multi-agent systems, centralized approaches always fail when the number of agents increases. For example, centralized shielding for MARL fails empirically for two-agent scenarios [18]. Fully decentralized shielding separates the whole state space into exclusive subspaces and synthesizes a shield to monitor a subspace. For example, factored shielding [18] computes multiple shields based on a factorization of the joint state space observed by all agents. However, this approach causes conservative behaviors (i.e., agents stuck in place) when agents move across the border of shields due to the information isolation between shields. Specifically, as shown in Figure 3, the shield would reject agents' actions even for those that are essentially valid. Consequently, the MARL system has higher coordination overhead, which causes extra steps when agents interact and render the MARL policy sub-optimal. In Section 6, we empirically demonstrate that the coordination overhead caused by conservative behaviors leads to sub-optimal policies.

4.2 Dynamic shielding

To mitigate the coordination overhead caused by conservative behaviors, we propose *dynamic shielding*, a decentralized shield framework on top of the traditional MARL process. Dynamic shielding



(a) Learn control policies with the protection of Dynamic Shielding.

(b) Learn Dynamics from experience.

Figure 4: Dynamic Shielding Framework

Algorithm 1: Dynamic Shielding at timestep t

Initialize : A list of shields $S = \{s_1, s_2, ..., s_m\}$, MARL agents' joint action $a_t = (a_t^1, a_t^2, ...a_t^n)$ and joint state $s_t = (s_t^1, s_t^2, ..., s_t^n)$, a constant penalty for unsafe actions p, a environment dynamics model $p(s_t|s_{t-1}, a_{t-1})$

Output : Safe joint action $\bar{a}_t$, punishment p_t, shield new_shield

1 // Clustering: divide agents into groups
2 // e.g., cluster agents by their position
3 *new_shield* = *cluster_agents*(s_t, a_t)
4 **for** *all group in* $i \in \{1, ..., m'\}$ **do**
5 **for** *all shield* $j \in \{, ..., m\}$ **do**
6 **if** *new_shield[I].group == s_j.group*
7 *and* s_j*.duration* ! = *0* **then**
8 new_shield[i].recompute = False
9 new_shield[i].shield = s_j*.shield*
10 **end if**
11 **end for**
12 **end for**
13 // Re-construct shields
14 // 1. When agents trying to escape shields
15 // 2. When shields expire
16 **for** *all group in* $i \in \{1, ..., m\}$ **do**
17 **if** *new_shield*$[i]$*.recompute* == *True* **then**
18 new_shield[i].reCompute(p)
19 **end if**
20 **end for**
21 // Shielding: Replace unsafe actions to safe actions
22 $\bar{a}_t$ = safe action output by new_shield
23 **for** *all agent in* $i \in \{1, ..., n\}$ **do**
24 **if** $\bar{a}_t^i \neq a_t^i$ **then**
25 p_t^i = p
26 **end if**
27 **end for**
28 **return** $bara_t$, p_t, *new_shield*

has two important features: 1) Dynamic shielding dynamically constructs shields based on agents' real-time states; 2) Dynamic shielding can perform two important operations, namely, *merge* and *split*. The merge operation uses multiple shields' information to construct a larger shield, which temporarily removes the border between shields. Therefore, the merged shield has enough information to distinguish whether joint actions are safe and eventually mitigate conservative behavior. On the other hand, the computation complexity in shield synthesis increases along with the shield size. The split operation helps decrease computation costs when agents locate sparsely. Figure 4a shows the diagram of dynamic shield construction. Initially, we construct distinct shields for each agent, which monitor agents' reachable states in the next k steps. If agents try to move to states outside the shield, the shield will recompute to establish a monitor on agents' future possible states. When agents gathering together has the possibility of collision, shields will merge to jointly monitor the action using the state information of multiple agents. When agents are more sparse, the merged shield will split to save computation.

We summarize dynamic shielding in algorithm 1. There are three phases: 1) clustering, 2) shield reconstruction, and 3) shielding. In the clustering phase (LINE 1-12), the algorithm clusters agents into groups by their current state. For example, in robot navigation tasks, if some agents are close by, the algorithm will put them in the same group, otherwise in separate groups. Agents in the same group should merge shields to avoid conservative behaviors. Then, in the shield re-construction phase (LINE 13-20), shields will merge with other shields or split into multiple smaller shields based on the results of clustering. In addition, some expired shields might recompute according to agents' state change. The merge operation could be implemented by recompute shield using agents' aggregated state information. In the shielding phase (Lines 21-27), every shield will do shielding concurrently, which rejects agents' unsafe actions and replaces them with safety actions. Lastly, the MARL agents will be given an extra penalty for unsafe actions.

Our method faces the challenge it degrades to centralized shielding for edge scenarios (some timesteps). For example, when all agents gather together, all decentralized shields will merge together into a single centralized shield. We propose an online method of shield synthesis in Section 5, which could efficiently synthesize shields.

Algorithm 2: Learn dynamics model

Initialize: Initialize dataset $\mathcal{D}$.
Initialize: Initialize neural network parameters θ randomly.
while *not converged* **do**
  // Dynamics Learning
  for *update step* $c = 1...C$ **do**
    Draw B data sequences $(o_t, a_t)_{t=k}^{k+L}$
    Update θ. // $p_\theta(o_t \mid o_{t-1}, a_t)$
  end for
  // Collecting Data
  o_1 = env. reset()
  while *episode not stopped* **do**
    $a_t \sim A$
    $o_{t+1} \leftarrow$ env. step(o_t, a_t)
  end while
  Add experience to dataset $\mathcal{D}$
end while
return θ

5 SYNTHESIZE SHIELD IN REAL-TIME

In this section, we introduce the incorporation of world model learning and present our shield synthesis method – *k-step look ahead shields*, a variant of traditional shield synthesis [27]. We also give theoretical proof to show that our method guarantees safety.

5.1 Learn the environment dynamics

To cope with the scenario that the MARL agents do not have external knowledge about the environment in the first place, our framework will train a coarse world model at the beginning. This world model is a deep neural network that learns to predict the environmental dynamics related to safety considerations. For example, an autonomous driving car could have different sensor inputs such as Lidar, Cameras, and GPS. If we only care about safety related to locomotion, the coarse world model will be trained to predict GPS signals under control inputs. There are many existing works on world model learning [20–22]. We adapt a general framework from Recurrent State-Space Model (RSSM) [21], which consists of three components (Fig 4b): encoder, decoder, and dynamics networks, which are denoted by $\text{enc}_\theta(s_t \mid s_{t-1}, x_t)$, $\text{dec}_\theta(s_t) \approx x_t$, and $\text{dyn}_\theta(s_t \mid s_{t-1}, a_{t-1})$ respectively.

With such a model obtained by Algorithm 2, given state x_t and action a_t with latent of previous state s_{t-1}, we can predict the next state x_{t+1} by:

$$\begin{aligned} s_t &\sim \text{enc}_\theta(s_t \mid s_{t-1}, x_t) \\ s_{t+1} &\sim \text{dyn}_\theta(s_{t+1} \mid s_t, a_t) \\ x_{t+1} &\sim \text{dec}_\theta(s_{t+1}) \end{aligned}$$

In this work, we focus on using the world model to learn the low-dimensional intrinsic properties of the environment, such as physical dynamics, for shield synthesis. We assume the existence of an expressive world model, which allows us to abstract away from the details of the sensory input and reason about the environment at a higher level of abstraction. For this work, the cascading error that may arise from errors in the learned world model is outside the scope of our discussion.

5.2 k-step look ahead shields

We assume the state space has been converted into a symbolic abstraction (via $f : S \to L$) given by a DFA $\mathcal{A}^e = \left(Q^e, q_0^e, \Sigma^e, \delta^e, F^e\right)$. We translate the LTL safety specification into another DFA $\mathcal{A}^S = \left(Q^S, q_0^S, \Sigma^S, \delta^S, F^S\right)$. We formulate a two-player game

$$\mathcal{G} = (G, g_0, \Sigma_1, \Sigma_2, \delta^g, F)$$

by combining $\mathcal{A}^e$ and $\mathcal{A}^S$. Instead of solving the game G directly, we add extra time constraints $t \le k$, where $t \in$ denotes the time step from constructing the shield, and k is a hyper-parameter that denotes the maximum steps of the game. The modified game is then

$$\mathcal{G}^k = \left(G^k, g_0{}', \Sigma_1, \Sigma_2, \delta^{g\prime}, F^k\right) \tag{1}$$

where the state space $G^k = G \times \{1...k\}$, the initial state $g_0{}' = (g_0, t = 1)$, the transition function $\delta^{g\prime}(g_t, t) = (\delta^g(g_t), t+1)$, which could be approximated through the world model, and the winning condition $F^k = F \wedge (t \le k)$. We can solve the two-player safety game $\mathcal{G}^k$ and compute the winning region $W \subseteq F^k$, using the method in [27]. We then construct the *k-step look ahead shield* by translating $\mathcal{G}^k$ and W to a reactive system $S = (Q_\mathcal{S}, q_{0,\mathcal{S}}, \Sigma_{I,\mathcal{S}}, \Sigma_{O,\mathcal{S}}, \delta_\mathcal{S}, \lambda_\mathcal{S})$. The shield has the following components: $Q_\mathcal{S} = G^k$, $q_{0,\mathcal{S}} = q_0{}'$, $\Sigma_{I,\mathcal{S}} = L \times \mathcal{A}$, $\Sigma_{O,\mathcal{S}} = \mathcal{A}$, $\delta_\mathcal{S}(g^k, (l, a)) = \delta\left(g^k, (l, \lambda_\mathcal{S}(g, (l, a)))\right)$ for all $g^k \in G, l \in L, a \in \mathcal{A}$, and

$$\lambda_\mathcal{S}(g, l, a) = \begin{cases} a & \text{if } \delta^k(g^k, (l, a)) \in W \\ a' & \text{if } \delta^k(g^k, (l, a)) \notin W \text{ for some arbitrary} \\ & \text{default } a' \text{ with } \delta^k\left(g^k, (l, a')\right) \in W. \end{cases}$$

Our shield synthesis bears a resemblance to the classic shield synthesis [1, 18], which also synthesizes shields by solving the two-player game. The main difference is that our method only predicts a subset of future state space, whereas previous methods enumerate all possible states along the planning horizon. This leads to the major benefit of our method, that for tasks with state spaces too large to compute in advance, our algorithm still works efficiently while previous methods fail.

5.3 Safety Guarantee

We show that dynamic shielding with *k-step look ahead shielding* can guarantee safety for MARL agents.

Proposition 1. *Given a trace $s_0 a_0 s_1 a_1 \cdots \in (S \times A)^\omega$ jointly produced by MARL agents, the dynamic shielding, and the environment, state-action pair (s_t, a_t) is safe at every time step regarding definition 1.*

Proof. Firstly, the procedure in algorithm 1 ensures each agent is monitored by a shield at each time step, and this shield at least monitors the states of agents in the next k steps (otherwise, the shield will re-compute). Then the remaining proof is the same as the correctness of centralized shielding in [18]. For any agents under shield $\mathcal{S} = (Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$, there is a corresponding

run $q_0 q_1, ...q_m \in (S \times A)^\omega$, where $m \leq k$ is the duration before reconstructing this shield. By constructing the shield, we have the environment abstraction DFA A^e and the safety specification DFA A^s. We can project the run $q_0, q_1, ...q_m$ of the shield $\mathcal{S}$ onto a trace $q_0^s(f(s_0), a_0)q_1^s(f(s_1), a_1)...q_m^s(f(s_m), a_m)$ on A_s. Since we construct the shield from the winning region of the two-player safety game, every state $q_i^s(f(s_i)$ visited by agents along the trace should be a safe state in $\mathcal{A}^s$. The shield $\mathcal{S}$ ensures the safety specification defined in $\mathcal{A}^s$ is never violated. Therefore, the joint state-action pair (s_t, a_t) is safe for every MARL agent at every step. □

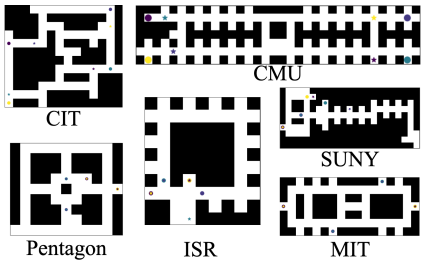


Figure 5: Different gridworld environments. Dots are agents, stars denote targets, and black blocks are obstacles.

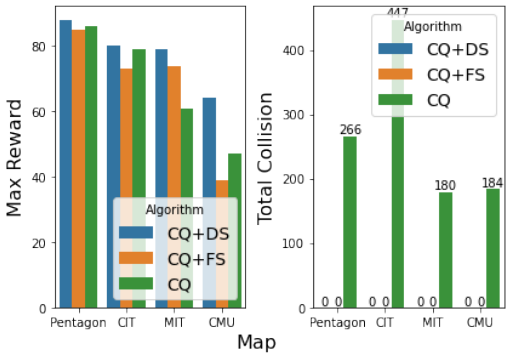
(a) Simple Spread **(b) Simple Adversary**

Figure 6: Modified Multi-Agent Particle Environment (MPE). The blue circles denote agents, black dots are landmarks. In *Simple Adversary*, the red circle is an adversary agent, and the green dot is the target landmark, we let $n_{good} : n_{adversary} = 3 : 1$. The MPE environment is unbounded, but agents will be penalized if they move too far away.

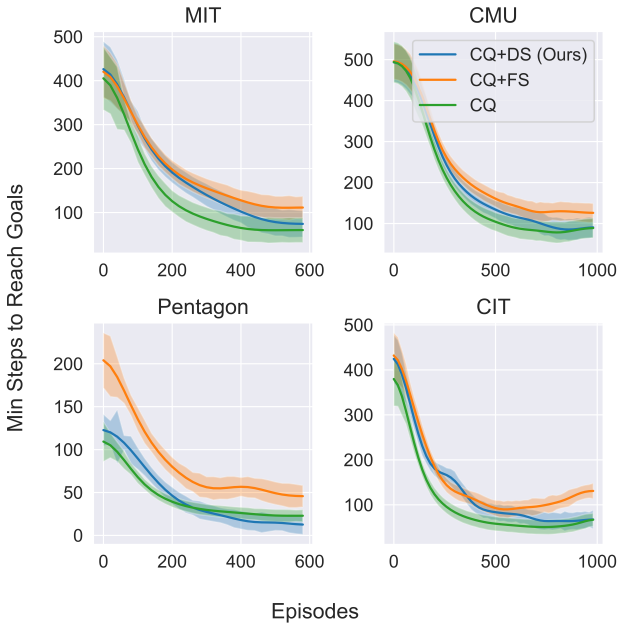
6 EXPERIMENTS

In this section, we empirically evaluate the performance of the proposed safe MARL framework (Algorithm 1) on multiple benchmark tasks. We apply our algorithm to four different maps of the *gridworld* [39] (shown in Figure 5) and two environments (cooperative and mixed-cooperative of MPE (shown in Figure 6). We compare the proposed algorithm with CQ-Learning [16], CQ with factored shielding (CQ+FS), DDPG [31], MADDPG [32], and MADDPG+CBF [11]. We conducted three sets of experiments in total. For the first two sets, we assume known environment dynamics to evaluate the performance of our shielding framework conveniently.

In the last experiment, we train our framework without any external knowledge of the environment to demonstrate the effectiveness of MARL with the proposed shielding in practice. We implement algorithms using Python and synthesize shields via Slugs [17]. For each experiment, we evaluate algorithms in both the training and testing phases. To mitigate outliers, we performed all experiments in 5 independent runs and averaged the results.



(a) Left-hand side figure is the max reward that agents obtained during learning. The right-hand side figure is the total collision between agents during learning.



(b) The *Min step to reach goals* value is an indicator of the optimality of learned policies.

Figure 7: Gridworld experiment. *CQ+DS*, *CQ+FS*, and *CQ* denote CQ-Learning with dynamic shielding, CQ-Learning with factored shielding, and CQ-Learning without shielding.

Experiment Setup. Figure 5 shows six maps of grid world benchmark environments adapted from the *gridworld* [39]. Each map has four agents learning to navigate while avoiding obstacles in the environment. The action set is $\mathcal{A} = \{stay, up, down, left, right\}$.

We randomly assign a unique target to each agent. Once an agent reaches its target, it stays there until all agents reach their goals. We set sparse rewards for this task, namely, giving a −1 living penalty, −10 collision penalty, and +100 for reaching the target.

Figure 6 shows two tasks from the modified MPE [32], say *simple spread* and *simple adversary*. The goal of *simple spread* task is for agents to cooperate and reach their target while avoiding collisions. The goal of *simple adversary* task is for good agents to navigate to the target and trick the adversary, and the adversary agents try to reach the target while avoiding collisions. These tasks are more difficult than the *gridworld* in two aspects:

(1) The state space of MPE is continuous and unbounded.
(2) Agents have more complicated dynamics in MPE, such as momentum and acceleration.

The action set is $\mathcal{A}^{'} = \{stay, up, down, left, right, \textbf{brake}\}$, from which the action controls acceleration. For example, if an agent takes *stay*, it moves at its original velocity instead of staying. We use *brake* to simulate braking in the real world, where the agent exerts a large deceleration in the direction of velocity until it stops. The *brake* action obeys the law of kinematics; for example, an agent moving at a higher speed needs a longer distance to brake down. Each agent receives a reward that is inversely proportional to the distance with its target and penalties for collisions.

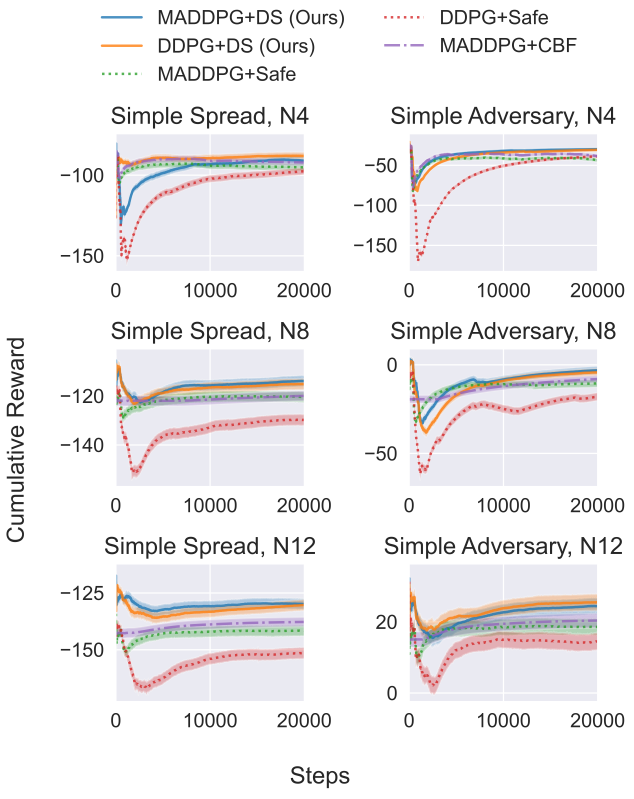


Figure 8: Compare Dynamic Shielding (MADDPG/DDPG+DS) with other baselines (MADDPG/DDPG+Safe, MADDPG+CBF).

Conservative Behavior Evaluation. We integrate CQ-learning with factored shielding and proposed dynamic shielding. We apply them to four different *gridworld* environments (Figure 5). We evaluate algorithms using maximum rewards, collision counts, and episode steps during the training phase. Results in Figure 7a show that both factored shielding and dynamic shielding can guarantee collision-free learning in all maps. However, dynamic shielding obtains better policies with higher rewards compared to factored shielding and learning without shielding. Figure 7b shows CQ+DS agents need fewer steps to reach the target than CQ+FS. The dynamic shielding policy eventually has comparable performance as CQ-learning without intervention. Therefore, that demonstrates the proposed dynamic shielding mitigates coordination overheads caused by factored shielding with the guarantee of safety.

Scalability Evaluation. We evaluate the performance of the dynamic shielding when the state space and the number of agents scale up. We integrate DDPG and MADDPG with dynamic shielding and apply them to the modified MPE environment (shown in Figure 6). The state space of MPE is a scale-up of gridworld as we described previously. Factored shielding fails in this unbounded environment since we cannot synthesize shields for the entire state space beforehand. We consider two baseline algorithms that incorporate safety mechanisms into DDPG/MADDPG: DDPG/MADDPG+Safe, which adds safety rewards to the reward function, and MADDPG+CBF, which enforces safety using control barrier functions. For MADDPG+CBF, we follow the barrier functions proposed in [11]. Table 1 shows MADDPG/DDPG+DS and MADDPG+CBF guarantee collision-free regardless of the tasks. Whereas, MADDPG/DDPG+Safe constantly have collisions, which increases as the number of agents scales up. Table 1 also depicts the cumulative rewards during the testing phase. At convergence, MADDPG/DDPG+DS obtains higher rewards than MADDPG/DDPG+Safe. The learning curves in Figure 8 demonstrate that the MARL algorithms with dynamic shielding converge faster at higher rewards. We also evaluate the performance of dynamic shielding when the number of agents scales up. Table 1 shows that although the collision of MADDPG/DDPG+Safe increase as agents scales up, MADDPG+DS and MADDPG+CBF always ensures safety and has higher rewards in both environments. As the number of agents increases, we observe a widening performance gap between MADDPG+DS and MADDPG+CBF. This suggests that our dynamic shielding still maintains minimal intervention (less conservative) as the number of agents scales up.

Model Based Dynamic Shielding. In this experiment, we remove the *brake* action in the environment and evaluate algorithms in the standard MPE *Simple spread* and *Simple Adversary* environments. Agents collect $3e5$ roll-outs from the environment to train the world model via Algorithm 2 to learn to predict next step location x_{t+1} based on current location, velocity, and action $[x_t, v_t, a_t]^T$. Since the temporal information is irrelevant to the dynamics in this environment, we use a $32 \times 64 \times 32$ MLP network as the dynamics model. We name this procedure Model-Based Dynamic Shielding (MBDS). We calculate testing phase safety rate by

$$r_{safety} = \frac{\sum_i \mathbb{1}(\text{collisions in step i} > 0)}{\text{number of steps}}.$$

Figure 9 demonstrates that MARL with Model-Based Dynamic Shielding (MADDPG+MBDS, DDPG+MBDS) obtains at least not

		MADDPG+DS		MADDPG+Safe		DDPG+DS		DDPG+Safe		MADDPG+CBF	
Task	N	REW	COL	REW	COL	REW	COL	REW	COL	REW	COL
Spread	4	-77 ± 8	**0.0**	-84 ± 6	1.3 ± 0.9	-76 ± 8	**0.0**	-82 ± 6	0.9 ± 1.0	-75 ± 6	**0.0**
	8	-112 ± 10	**0.0**	-119 ± 9	10.0 ± 2.2	-113 ± 10	**0.0**	-125 ± 13	7.7 ± 1.9	-118 ± 12	**0.0**
	12	-129 ± 9	**0.0**	-141 ± 9	38.2 ± 6.6	-129 ± 11	**0.0**	-151 ± 14	26.5 ± 6.5	-138 ± 11	**0.0**
Adversary	4	-25 ± 3	**0.0**	-30 ± 4	1.1 ± 0.8	-26 ± 3	**0.0**	-26 ± 3	0.9 ± 1.3	-23 ± 3	**0.0**
	8	-1 ± 3	**0.0**	-5 ± 8	6.8 ± 2.0	-2 ± 4	**0.0**	-11 ± 16	5.5 ± 1.7	-2 ± 5	**0.0**
	12	21 ± 9	**0.0**	16 ± 8	43.5 ± 10.6	23 ± 9	**0.0**	13 ± 11	33.2 ± 10.9	18 ± 8	**0.0**

Table 1: Results comparing the average rewards and collisions of algorithms during the testing phase. In the table, $a \pm b$ denotes the mean and variance of results from 10 independent testing runs. (*REW, COL* denote cumulative rewards and collisions).

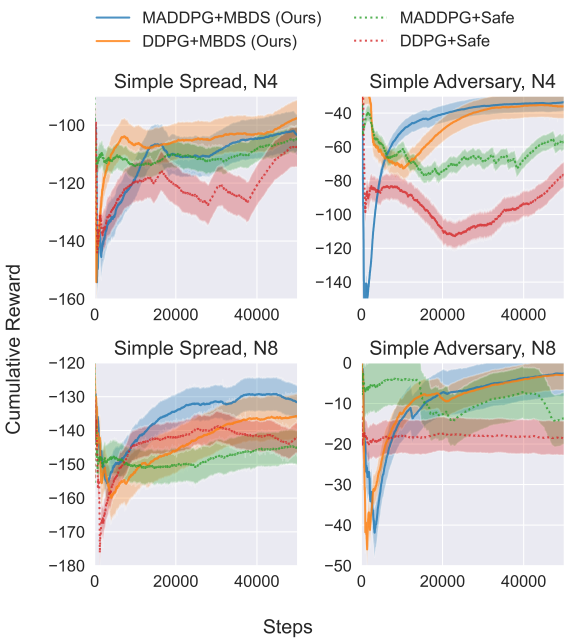


Figure 9: Compare Model-Based Dynamic Shielding (MBDS) with other baselines.

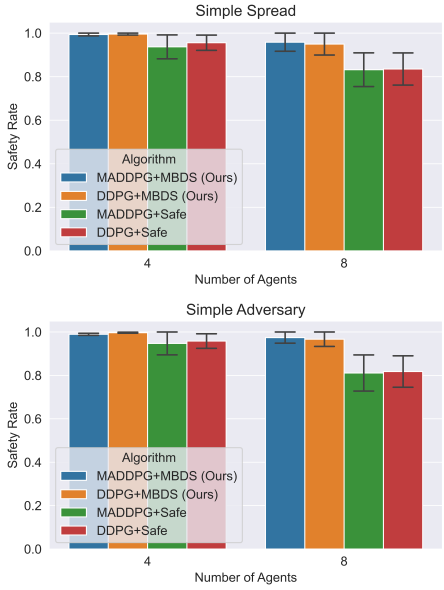


Figure 10: Safety rate in the MPE tasks.

lower cumulative rewards than other baselines (MADDPG+Safe, DDPG+Safe). In addition, when the number of agents increases from 4 to 8, the safety rates of shielding (shown in Figure 10) decrease no more than 5% and keep above 90%. However, other baselines has safety rates decrease 10% in *simple spread* and 20% in *simple adversary*. Hence, Model-Based Dynamic Shielding is an effective method to provide safety guarantees for MARL.

7 CONCLUSIONS

This paper presents a novel approach to addressing safe MARL through model-based dynamic shielding. The proposed method minimally interferes with the MARL framework to ensure the safety specification defined by LTL expressions. We also propose an effective technique to synthesize shields in real time and provide theoretical proof of a safety guarantee. In addition, we conduct extensive experiments to demonstrate our algorithm is better than other baselines regarding safety and learning performance in benchmark tasks. There are some limitations that we acknowledge. First, our approach does not address the issue of cascading errors that may arise from inaccuracies in the learned dynamics model. This could potentially impair the safety guarantees provided by our framework. Second, we have not analyze the time complexity of our algorithm, which is heavily dependent on the implementation of the reactive game solver. In our future work, we plan to explore methods for enforcing safety guarantees in the presence of a risk-aware dynamics model, which would help to mitigate the impact of cascading errors. Additionally, we will conduct a thorough analysis of the time complexity of our algorithm to ensure that it can scale to larger and more complex environments.

ACKNOWLEDGMENTS

We thank Ingy ElSayed-Aly and Daniel Melcer for the insightful discussion about this work. The work was conducted as part of an undergraduate research experience with the Carnegie Mellon University Robotics Institute Summer Scholars Program. The scholarship funding for Wenli Xiao was provided by the Shenzhen Institute of Artificial Intelligence and Robotics for Society.

REFERENCES

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[2] Rajeev Alur. 2015. *Principles of cyber-physical systems.* MIT press.

[3] Itamar Arel, Cong Liu, Tom Urbanik, and Airton G Kohls. 2010. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems* 4, 2 (2010), 128–135.

[4] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking.* MIT press.

[5] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking.* MIT press.

[6] Osbert Bastani, Shuo Li, and Anton Xu. 2021. Safe Reinforcement Learning via Statistical Model Predictive Shielding.. In *Robotics: Science and Systems*. 1–13.

[7] Sushrut Bhalla, Sriram Ganapathi Subramanian, and Mark Crowley. 2020. Deep multi agent reinforcement learning for autonomous driving. In *Canadian Conference on Artificial Intelligence*. Springer, 67–78.

[8] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. 2015. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine* 48, 27 (2015), 68–73.

[9] Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. 2020. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10349–10355.

[10] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[11] Zhiyuan Cai, Huanhui Cao, Wenjie Lu, Lin Zhang, and Hao Xiong. 2021. Safe multi-agent reinforcement learning through decentralized multiple control barrier functions. *arXiv preprint arXiv:2103.12553* (2021).

[12] Yuxiao Chen, Huei Peng, and Jessy Grizzle. 2017. Obstacle avoidance for low-speed autonomous vehicles with barrier function. *IEEE Transactions on Control Systems Technology* 26, 1 (2017), 194–206.

[13] Yuxiao Chen, Andrew Singletary, and Aaron D Ames. 2020. Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach. *IEEE Control Systems Letters* 5, 1 (2020), 127–132.

[14] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. 2019. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3387–3395.

[15] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. 2018. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems* 31 (2018).

[16] Yann-Michaël De Hauwere, Peter Vrancx, and Ann Nowé. 2010. Learning multi-agent state space representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. 715–722.

[17] Rüdiger Ehlers and Vasumathi Raman. 2016. Slugs: Extensible gr (1) synthesis. In *International Conference on Computer Aided Verification*. Springer, 333–339.

[18] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. 2021. Safe multi-agent reinforcement learning via shielding. *arXiv preprint arXiv:2101.11196* (2021).

[19] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[20] David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122* (2018).

[21] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. Learning latent dynamics for planning from pixels. In *International conference on machine learning*. PMLR, 2555–2565.

[22] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. Learning latent dynamics for planning from pixels. In *International conference on machine learning*. PMLR, 2555–2565.

[23] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2019. Omega-regular objectives in model-free reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 395–412.

[24] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2020. Cautious reinforcement learning with logical constraints. *arXiv preprint arXiv:2002.12156* (2020).

[25] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2020. Cautious reinforcement learning with logical constraints. *arXiv preprint arXiv:2002.12156* (2020).

[26] Nils Jansen, Bettina Könighofer, JSL Junges, AC Serban, and Roderick Bloem. 2020. Safe reinforcement learning using probabilistic shields. (2020).

[27] Bettina Könighofer, Mohammed Alshiekh, Roderick Bloem, Laura Humphrey, Robert Könighofer, Ufuk Topcu, and Chao Wang. 2017. Shield synthesis. *Formal Methods in System Design* 51, 2 (2017), 332–361.

[28] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. 2009. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics* 25, 6 (2009), 1370–1381.

[29] Orna Kupferman and Moshe Y Vardi. 2001. Model checking of safety properties. *Formal methods in system design* 19, 3 (2001), 291–314.

[30] Shuo Li and Osbert Bastani. 2020. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7166–7172.

[31] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[32] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)* (2017).

[33] Songtao Lu, Kaiqing Zhang, Tianyi Chen, Tamer Başar, and Lior Horesh. 2021. Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8767–8775.

[34] Wenhao Luo, Wen Sun, and Ashish Kapoor. 2020. Multi-robot collision avoidance under uncertainty with probabilistic safety barrier certificates. *Advances in Neural Information Processing Systems* 33 (2020), 372–383.

[35] Yiwei Lyu, Wenhao Luo, and John M Dolan. 2021. Probabilistic safety-assured adaptive merging control for autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10764–10770.

[36] Yiwei Lyu, Wenhao Luo, and John M Dolan. 2022. Adaptive safe merging control for heterogeneous autonomous vehicles using parametric control barrier functions. In *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 542–547.

[37] Yiwei Lyu, Wenhao Luo, and John M Dolan. 2022. Responsibility-associated Multi-agent Collision Avoidance with Social Preferences. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 3645–3651.

[38] Francisco S Melo and Manuela Veloso. 2009. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. 773–780.

[39] Francisco S Melo and Manuela Veloso. 2009. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. Citeseer, 773–780.

[40] Liane Okdinawati, Togar M Simatupang, and Yos Sunitiyoso. 2017. Multi-agent reinforcement learning for value co-creation of collaborative transportation management (CTM). *International Journal of Information Systems and Supply Chain Management (IJISSCM)* 10, 3 (2017), 84–95.

[41] Martin Pecka and Tomas Svoboda. 2014. Safe exploration techniques for reinforcement learning–an overview. In *International Workshop on Modelling and Simulation for Autonomous Systems*. Springer, 357–375.

[42] Adolfo Perrusquía, Wen Yu, and Xiaoou Li. 2020. Redundant robot control using multi agent reinforcement learning. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 1650–1655.

[43] Amir Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. ieee, 46–57.

[44] Stephen Prajna, Ali Jadbabaie, and George J Pappas. 2007. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Automat. Control* 52, 8 (2007), 1415–1428.

[45] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. 2021. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436* (2021).

[46] Kristin Y Rozier. 2011. Linear temporal logic symbolic model checking. *Computer Science Review* 5, 2 (2011), 163–203.

[47] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295* (2016).

[48] Mohit Srinivasan and Samuel Coogan. 2020. Control of mobile robots using barrier functions under temporal logic specifications. *IEEE Transactions on Robotics* 37, 2 (2020), 363–374.

[49] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. 2020. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*. PMLR, 708–717.

[50] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. 2021. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters* 6, 3 (2021), 4915–4922.

[51] Alphan Ulusoy, Stephen L Smith, Xu Chu Ding, Calin Belta, and Daniela Rus. 2013. Optimality and robustness in multi-robot path planning with temporal logic constraints. *The International Journal of Robotics Research* 32, 8 (2013), 889–911.

[52] Spencer Van Koevering, Yiwei Lyu, Wenhao Luo, and John Dolan. 2022. Provable Probabilistic Safety and Feasibility-Assured Control for Autonomous Vehicles using Exponential Control Barrier Functions. In *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 952–957.

[53] Akifumi Wachi and Yanan Sui. 2020. Safe reinforcement learning in constrained Markov decision processes. In *International Conference on Machine Learning*. PMLR, 9797–9806.

[54] Akifumi Wachi and Yanan Sui. 2020. Safe reinforcement learning in constrained Markov decision processes. In *International Conference on Machine Learning*. PMLR, 9797–9806.

[55] Li Wang, Aaron D Ames, and Magnus Egerstedt. 2017. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics* 33, 3 (2017), 661–674.

[56] Peter Wieland and Frank Allgöwer. 2007. Constructive safety using control barrier functions. *IFAC Proceedings Volumes* 40, 12 (2007), 462–467.

[57] Chao Yu, Xin Wang, and Zhanbo Feng. 2019. Coordinated multiagent reinforcement learning for teams of mobile sensing robots. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems*. 2297–2299.

[58] Jun Zeng, Bike Zhang, and Koushil Sreenath. 2021. Safety-critical model predictive control with discrete-time control barrier function. In *2021 American Control Conference (ACC)*. IEEE, 3882–3889.

[59] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.

[60] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. 2020. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776* (2020).